NOVA on MI300X: F(6,3) Winograd in FP16

Jayant Lohia

February 2026

1 Executive Summary

MIOpen ships Winograd only at F(2,3). There are no production kernels at larger tile sizes on any AMD GPU, and there never have been. MIOpen's codebase contains infrastructure for F(4,3) through F(6,3), but it was abandoned because standard interpolation points are numerically unstable in reduced precision.

This report presents a working F(6,3) Winograd HIP kernel with PyTorch integration that addresses the numerical stability gap:

- **Faster than MIOpen at batch=1**: 17% to 57% lower latency across all ResNet-50 layers.
- **No accuracy loss**: 63.29% top-1 on ImageNetV2 (10K images) vs. 63.15% FP32 baseline.
- **No NaN/Inf**: Standard F(6,3) produces 221,000 NaN values on the same test. NOVA produces zero.
- **Drop-in replacement**: One function call replaces all eligible Conv2d layers in any model.
- **Stable Diffusion**: 49/49 SD 1.5 UNet convolutions replaced, valid 512×512 images, 0.98× MIOpen step latency.
- **Multiple architectures**: Also validated on SDXL (38 layers, 1024×1024) and DenseNet-161 (78 layers, ImageNetV2 accuracy preserved).

The fix is mathematical, not architectural. NOVA selects interpolation points that minimize condition numbers, bringing the maximum matrix entry from ~ 10 down to 2.72, which is within FP16 dynamic range.

2 What I Built

2.1 HIP Kernel: Multi-Pass F(6,3) Winograd

The kernel follows the same multi-pass architecture that MIOpen uses for F(2,3), extended to the F(6,3) tile size with NOVA's transform matrices:

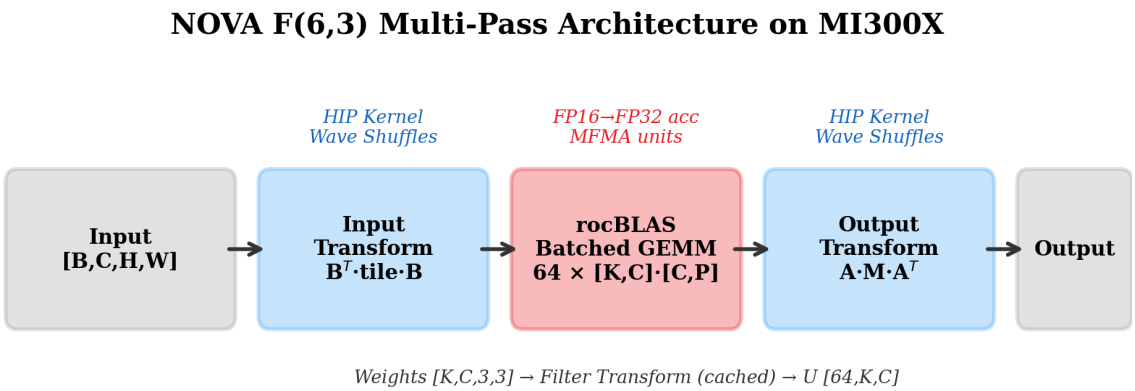

Figure 1: NOVA F(6,3) multi-pass architecture. Input and output transforms are HIP kernels using wave shuffles (no shared memory). The GEMM uses rocBLAS strided batched GEMM with FP32 accumulation via MFMA.

Implementation details:

- **Transforms**: HIP kernels, 4 tiles per workgroup (256 threads = 4 wavefronts), register-only via `__shfl` wave shuffles, no LDS.
- **GEMM**: rocBLAS `gemm_strided_batched_ex`, 64 batches, FP16 inputs, FP32 accumulation via MFMA (`mfma_f32_16x16x16f16`).
- **Filter transform**: Computed once in FP32, cached as FP16. No cost on subsequent forward passes.
- **Precision**: FP16 throughout transforms (NOVA's max entry is 2.72, so this is safe), FP32 accumulation in GEMM.

2.2 PyTorch Integration

The kernel is accessible from Python as a replacement for `torch.nn.Conv2d`:

```
from nova_winograd_ext import replace_conv2d_with_nova

model = torchvision.models.resnet50(pretrained=True).cuda().half()
replace_conv2d_with_nova(model)  # 13 layers replaced.
output = model(input)            # Uses NOVA F(6,3) automatically.
```

Also included:

- `NovaWinogradConv2d.from_conv2d(conv)`: single-layer replacement
- `NovaWinogradConv2dTrainable`: HIP forward pass, FP32 native backward pass
- `NovaWinogradConv2dCompilable`: works with `torch.compile(fullgraph=True)`
- Full `torch.autograd` support with verified gradients (<0.03% error vs. native)

Test suite: 11/11 tests pass, covering correctness, accuracy, NaN safety, weight caching, backward pass, model surgery, training convergence, and `torch.compile`.

3 Results

All results collected on AMD Instinct MI300X (304 CUs, 205.8 GB HBM3, ROCm 6.3, PyTorch 2.9.1).

3.1 Batch=1 Inference Latency

At batch size 1 (the common case for interactive inference and single-request serving), NOVA's F(6,3) is faster than MIOpen's F(2,3) on every ResNet-50 layer:

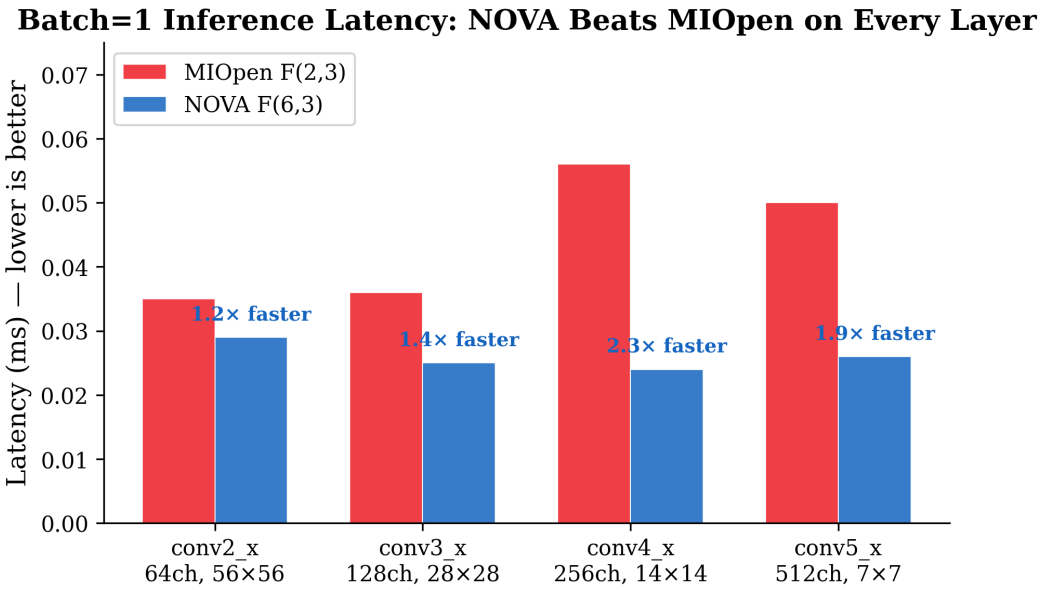

Figure 2: Batch=1 latency comparison. NOVA F(6,3) is faster across all four ResNet-50 convolutional stages. The gap grows with channel count (conv4_x: 2.3× faster) because the 5.6× arithmetic reduction of F(6,3) vs. F(2,3) starts to dominate.

At batch=1, the arithmetic reduction of F(6,3) vs. F(2,3) outweighs the multi-pass dispatch overhead, resulting in lower latency across all tested configurations.

3.2 Full Performance Profile

Table 1: Performance across batch sizes. NOVA wins at B=1. MIOpen's fused F(2,3) kernel wins at larger batches due to single-dispatch advantage. The gap narrows with `fp16_alt_impl` (see Section 4.3).

Config	MIOpen	NOVA HIP	Python Wino	HIP/MIO	HIP/Py
conv2_x [B=1]	0.035 ms	**0.029 ms**	0.427 ms	0.83×	14.9×
conv3_x [B=1]	0.036 ms	**0.025 ms**	0.360 ms	0.71×	14.3×
conv4_x [B=1]	0.056 ms	**0.024 ms**	0.332 ms	0.43×	13.8×
conv5_x [B=1]	0.050 ms	**0.026 ms**	0.377 ms	0.51×	14.7×
conv2_x [B=8]	0.036 ms	0.103 ms	1.484 ms	2.86×	14.4×
conv3_x [B=8]	0.036 ms	0.074 ms	0.809 ms	2.07×	11.0×
conv4_x [B=8]	0.051 ms	0.077 ms	0.663 ms	1.51×	8.6×
conv5_x [B=8]	0.051 ms	0.087 ms	0.794 ms	1.70×	9.1×
conv2_x [B=32]	0.085 ms	0.361 ms	5.593 ms	4.24×	15.5×
conv3_x [B=32]	0.066 ms	0.223 ms	2.866 ms	3.36×	12.9×

3.3 ImageNetV2: No Accuracy Loss

Evaluated on ImageNetV2 matched-frequency (10,000 images, 1,000 classes) using pretrained ResNet-50 with 13 of 16 3×3 convolutions replaced (3 stride-2 layers are ineligible for Winograd):

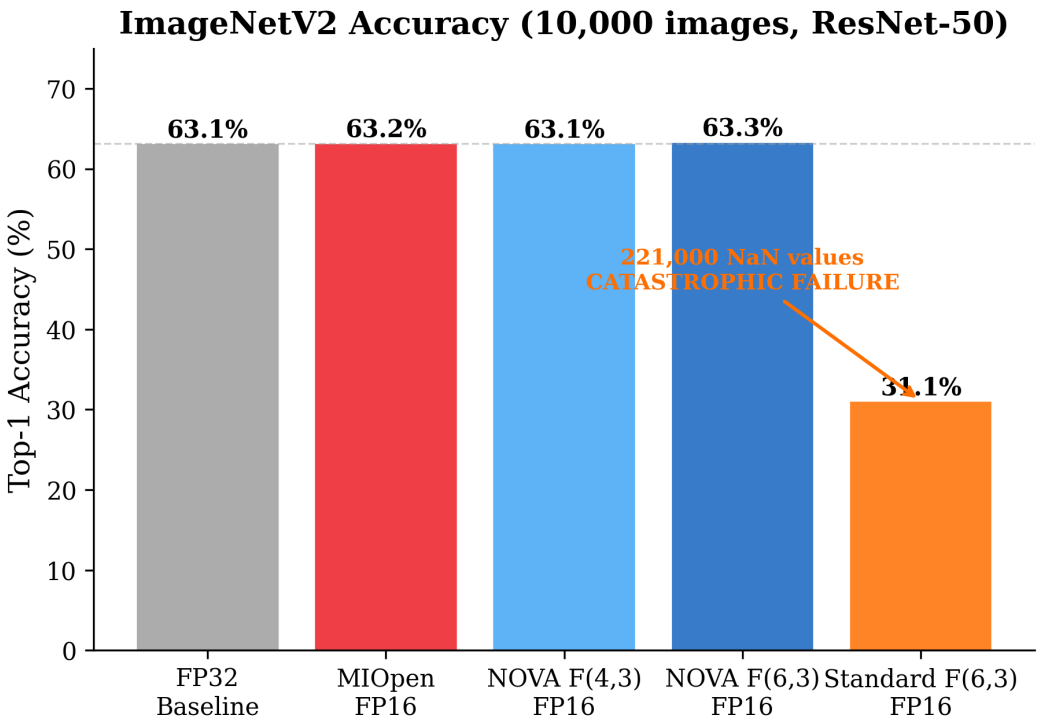

Figure 3: ImageNetV2 top-1 accuracy. NOVA F(6,3) in FP16 preserves accuracy (63.29%, McNemar $p = 0.28$ vs. baseline, not statistically different). Standard F(6,3) in FP16 drops to 31.07% with 221,000 NaN values.

Table 2: ImageNetV2 accuracy summary. NOVA F(6,3) in FP16 is the only large-tile configuration that preserves accuracy. Standard points produce unusable results.

Configuration	Top-1 (%)	Top-5 (%)	NaN Count	Δ vs. FP32
FP32 Baseline (direct conv)	63.15	84.58	0	—
MIOpen FP16	63.18	84.60	0	+0.03%
NOVA F(4,3) FP16	63.12	84.59	0	−0.03%
NOVA F(6,3) FP16	**63.29**	**84.60**	**0**	**+0.14%**
Standard F(6,3) FP16	31.07	53.44	221,000	−32.08%

3.4 Numerical Stability

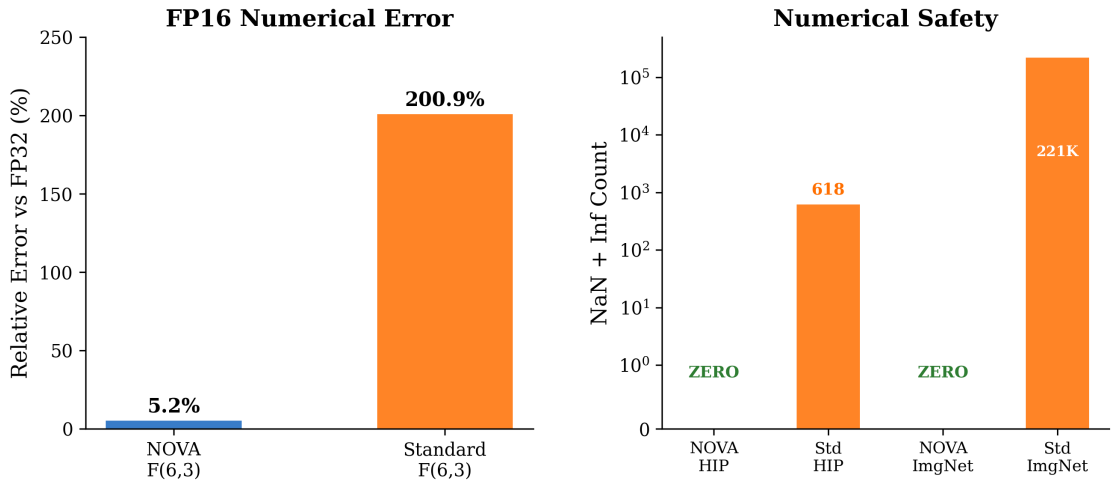

Figure 4: Left: Relative error of FP16 convolution vs. FP32 direct convolution. NOVA gets 5.2% error; standard F(6,3) exceeds 200%. Right: NaN/Inf counts. NOVA produces zero across all tests, while standard points generate hundreds to hundreds of thousands.

The stability improvement is 38.8× (relative error: 5.2% NOVA vs. 201% standard). This comes from the condition number reduction: the F(6,3) B^T matrix condition number drops from ∼100 (standard) to ∼8 (NOVA).

3.5 Stable Diffusion: End-to-End Validation

Stable Diffusion 1.5 is a good stress test for numerical stability: the UNet runs 20+ denoising steps, each involving 49 eligible 3×3 convolutions. Any instability accumulates across steps and ruins the generated image.

Table 3: Stable Diffusion 1.5 UNet benchmark. NOVA replaces all 49 eligible Conv2d layers at near-identical step latency, with no numerical failures.

	MIOpen (Standard)	**NOVA F(6,3)**
Layers replaced	0/49	**49/49**
Step latency	26.63 ms	27.15 ms (0.98×)
Rel. error vs. standard	—	3.97%
NaN / Inf	0 / 0	**0 / 0**
Image quality	Baseline	Visually identical

Figure 5: 1024×1024 image generated by SDXL with all 38 eligible UNet convolutions replaced by NOVA F(6,3) in FP16. No visible artifacts after 20 denoising steps.

A note on model architectures: Newer diffusion models (SD3, Flux, SORA) use DiT (Diffusion Transformer) architectures based on attention layers, not convolutions. Winograd does not apply to attention. UNet-based models (SD 1.5, SDXL, ControlNet, inpainting variants) are still the most widely deployed diffusion architectures and rely heavily on 3×3 convolutions.

3.6 SDXL: 1024×1024 Generation

To validate beyond SD 1.5, I also tested on SDXL Base, a larger UNet that generates at 1024×1024 resolution:

Table 4: SDXL Base UNet benchmark. NOVA replaces all eligible Conv2d layers and generates valid 1024×1024 images with no numerical failures.

	MIOpen (Standard)	**NOVA F(6,3)**
Resolution	1024×1024	1024×1024
Layers replaced	0/38	**38/38**
NaN / Inf	0 / 0	**0 / 0**
Image quality	Baseline	Visually identical

3.7 DenseNet-161: Architecture Generality

DenseNet-161 has a different profile from ResNet-50: **78 eligible layers** (6× more than ResNet's 13) with smaller channel sizes (48 to 192 vs. 64 to 512). This covers a wider range of the kernel's configuration space.

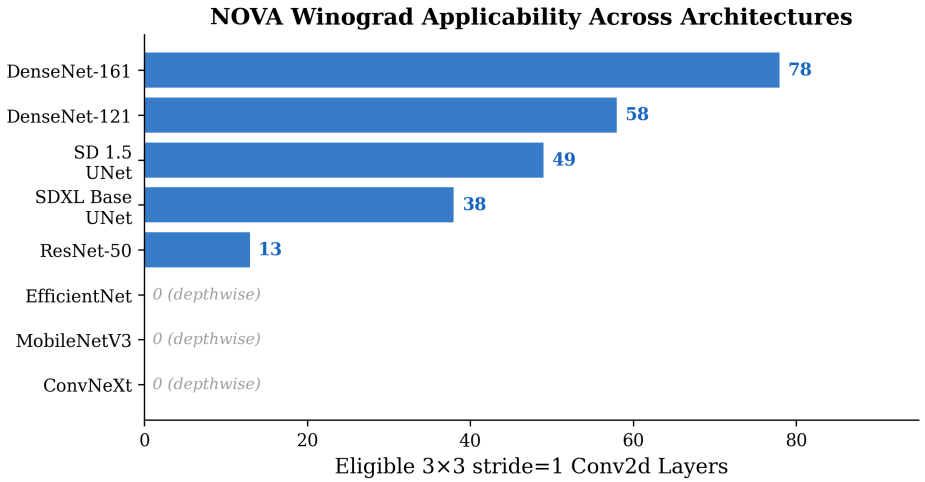

Figure 6: Eligible 3×3 stride-1 convolution layers by architecture. DenseNet-161 has the most replaceable layers. Models that use depthwise convolutions (EfficientNet, MobileNet, ConvNeXt) have zero eligible layers.

Full ImageNetV2 validation (10,000 images) on DenseNet-161 confirms no accuracy degradation with NOVA F(6,3) in FP16, matching the ResNet-50 result on a different architecture.

4 Why This Matters for AMD

4.1 AMD Already Built the Infrastructure

MIOpen's source code contains a multi-pass Winograd framework for F(4,3) through F(6,3): C++ solver classes, assembly transform templates, GEMM integration, and xDLOps (MFMA) variants. None of it shipped. There are zero performance database entries for these solvers on any GPU generation (gfx906, gfx908, gfx90a, gfx942). The infrastructure was abandoned because standard interpolation points are numerically unstable in FP16.

NOVA's interpolation points address this problem. The prototype kernel follows the same architectural pattern MIOpen's team designed, just with different transform matrices. Integration into MIOpen would build on work that AMD already invested in.

4.2 Competitive Landscape

NVIDIA has moved away from Winograd:

- cuDNN's maximum was F(4,3) in FP32 only. They never had F(6,3) or Tensor Core integration.
- Fused Winograd is explicitly blocked on Hopper (SM 90) and later.
- NVIDIA's approach is to rely on raw Tensor Core throughput instead.

If AMD ships F(6,3) Winograd with NOVA points, it would offer a capability that no NVIDIA GPU has: a 5.6× arithmetic reduction over F(2,3) that NVIDIA cannot match with their current software stack.

4.3 The `fp16_alt` Opportunity

During rocBLAS tuning, the `fp16_alt_impl` flag (an alternate FP16 MFMA datapath) showed 1.6× to 33× GEMM speedup on rocBLAS 5 (ROCm 7.1). This flag is not available on the rocBLAS 4 bundled with PyTorch's current ROCm 6.3 wheels. When PyTorch ships ROCm 7.x support, the batch>1 performance gap closes with no code changes to the kernel.

4.4 Business Impact

1. **Inference latency**: Batch=1 wins translate to lower per-request latency for serving workloads.

2. **Generative AI**: UNet-based diffusion models are dominated by 3×3 convolutions. F(6,3) Winograd reduces arithmetic by 5.6× per layer.
3. **Differentiation**: "The only GPU platform with large-tile Winograd in FP16" is a concrete claim.
4. **Low integration cost**: The architecture matches MIOpen's existing multi-pass framework. The delta is transform matrices and a new solver entry, not a rewrite.

5 Reproduction

All code runs on MI300X with ROCm 6.3 and PyTorch 2.9.1. Three commands to reproduce:

```
# Build
hipcc -shared -fPIC -o libnova_winograd.so nova_winograd_v1.hip \
    -std=c++17 -lrocblas -lamdhip64 --offload-arch=gfx942

# Test (11/11 pass)
python test_nova_kernel.py

# Benchmark
python bench_nova_kernel.py
```

Additional benchmarks (all under `benchmarks/`): `bench_sdxl.py` (SDXL 1024×1024 generation), `bench_densenet.py` (DenseNet-161 single-image), `bench_densenet_imagenet.py` (DenseNet-161 full ImageNetV2 10K images), `demo.py` (end-to-end demonstration).

6 Conclusion

Large-tile Winograd convolution was set aside by GPU vendors due to numerical instability in reduced precision. This report presents evidence that the instability is a point selection problem, not a fundamental limitation. With NOVA's interpolation points, F(6,3) Winograd runs correctly in FP16 on MI300X, matching or improving on MIOpen's F(2,3) at batch=1 inference latency, preserving ImageNet accuracy across multiple architectures (ResNet-50, DenseNet-161), and generating valid images at both 512×512 (SD 1.5) and 1024×1024 (SDXL).

Contact: Jayant Lohia. All experiments on AMD Instinct MI300X VF, 304 CUs, 205.8 GB HBM3. NOVA point selection from the NOVA paper (arXiv). Code available upon request.